Northeastern University - Seattle

Khoury College of Computer Sciences

CS6200 Information Retrieval Fall 2019

Lecture 9:
Retrieval Models – 2

Oct 28, 2019

Administrivia

- Assignment 3
  - Hope all of you have started on this assignment and have done a fair bit of the assignment.
  - Warning: We are seeing a lot of submissions with the wrong files included, and the required files missing. This causes needless extra work for everybody.
    Submitting the right files is part of learning to be responsible.
    Henceforth, submissions with wrong files will lose at least 1 mark.
  - Questions?

Administrivia

- Half-way point
  - Feedback?
  - What can we do better?
- Final Exam
  - Dec 9th, 2 hours long.
  - 20% ; closed book, closed notes, except for a one-page 'cheat-sheet' *hand-written by you*.

Quiz 6 Discussion .. 1

1. Write down a query
[catalina problems]
Did it return results that satisfied your information need?
Yes.

If yes, list the name of the new release, and name some problems it has. *
macOS 10.15 Catalina
Problems include:

a. Can't install macOS Catalina. You've unpacked the installer and now macOS Catalina install stuck on the initial reboot. All you can see is ...

b. macOS Catalina freezing upon reboot.

c. macOS Catalina apps not working.

d. macOS Catalina is running slow.

e. Problem with Apple ID settings.

Quiz 6 Discussion .. 2

1. Keep the first letter (in upper case).
2. Replace these letters with hyphens: a,e,i,o,u,y,h,w.
3. Replace the other letters by numbers as follows:
   1: b,f,p,v
   2: c,g,j,k,q,s,x,z
   3: d,t
   4: l
   5: m,n
   6: r
4. Delete adjacent repeats of a number.
5. Delete the hyphens.
6. Keep the first three numbers or pad out with zeros.

2. Soundex and issues

Which of the following variations of Yovanovitch will be treated as equivalent to the original by the Soundex algorithm:
(a) Yovanovich , (b) Yovonavitch , (c) Yovonovitch. …
what do the results of this test tell you about Soundex?

Yovanovitch → Y-1-5-1-32- → Y15132 → Y151

All 3 variants also come to Y151.

Soundex depends a lot of the first few characters, and vowels changes do not matter much.

3. Norvig's speller:

Give a few examples of the kinds of spelling errors his speller will **not** catch. Explain why these will not be caught by Norvig's speller. Then briefly list some ideas about how you can improve the speller to catch these kinds of errors as well.

Spelling errors with edit distance > 2: Norvig's code considers correction only for candidates with edit distance <= 2 away.

Or words that are proper English, but not in the right context. [lead <-> led, bear <.> bar ..]

Ways to improve: …

Quiz 6 Discussion .. 3

4. Context in search

Think of a search example where without context you may not get great results. Think of some context the search engine could get … that could help disambiguate your query and get you better results. Mention the query … [and] the problem … without context. … what type of context could help, and …

Search “turkey”, the problem is “turkey” can either mean country, bird, or food. The context of search history and time of year would help. E.g. near Thanksgiving, user is more likely to search turkey the food. If the user search a lot about politics or other countries, then Turkey the country might be what user wants to search.

Good examples from class. One about parents and children particularly striking!

Quiz 7 Discussion .. 1

1. Show score?

   Imagine that search engines provide the score for each result, along with the ranking they provide now. Do you think this would be useful?

   The score may be helpful for users to refine their queries if the best results have a relatively large distance from the query. Or to see when the quality of results has a big drop.

   Or may be too complex for users?

2. Idf

What is the idf contribution from a term that occurs:

- Exactly twice in each and every document in this whole collection?
- Exactly once in each and every odd-numbered document in this collection?
- Exactly once in each and every odd-numbered document in this collection and
- exactly twice in each and every even-numbered document in this collection?

idf for term 2x/document, and
1x per odd doc and 2x per even doc:
Both the same. log(N/N) = log(1) = 0.
No contribution to idf.
idf for term 1x per odd doc:
Doc Freq = N/2.
idf = log(N/(N/2)) = log(2) = 0.3010

Quiz 7 Discussion .. 2

3. Base difference.

How will the ranking of documents differ in the vector space tf.idf model of ranking, if you changed the idf log from base 10 to base 8 (octal)?

Scores diff, ranking same, no advantage of either base

4. Compute log-weighted tf from raw tf: (1 + log(tf)), unless it's 0, multiply that by idf, for each doc-term combination. See next slide.

Remember to compute the right numbers in Assignment 3.

5. ltc.ltn

Remember ddd.qqq
ltc for document

L: log

T: idf

C: cosine

ltn for query

L: log

T: idf

N: none

Quiz 7 Discussion … 3

raw-tf, df, idf data

Term	D1	D2	D3
Nikon	26	5	23
Canon	4	31	0
lenses	0	32	28
tripod	15	0	14

Term	df	idf
Nikon	18163	1.65
Canon	6722	2.08
lenses	19240	1.62
tripod	25232	1.5

We first compute the weighted term frequencies for each word

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

Term	D1	D2	D3
Nikon	2.41	1.70	2.36
Canon	1.60	2.49	0.00
lenses	0.00	2.51	2.45
tripod	2.18	0.00	2.15

Then we add multiply each weighted tf by the term's idf score

Nikon idf = 1.65, Nikon weighted tf for D1 = 2.41,
Nikon's tf-idf contribution for D1 = 2.41 * 1.65 = 3.98 etc.

Term	D1	D2	D3
Nikon	3.98	2.80	3.90
Canon	3.33	5.18	0.00
lenses	0.00	4.06	3.96
tripod	3.26	0.00	3.22

Quiz 7 Discussion .. 4

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$				

Overview

Language Models

- Query Likelihood Model
  - Smoothing (Jelinek-Mercer, Dirichlet)
- Relevance Model
  - Kullback-Leibler (KL) Divergence

Retrieval Models & Web Search

Machine Learning & IR (Ranking SVM)

Topic Models (LDA)

LANGUAGE MODELS

(Great) Expectations .. 1

Words are not really independent

Not even characters

What character do you expect after a ‘q’?

What about after a ‘t’?

When do you expect a ‘j’ after a ‘j’ ? Example?

What about a ‘w’ after a ‘w’ ?

(Great) Expectations .. 2

What words would you fill the blanks with?

- Internal _____ Service
- _____ Desk
- Privacy _____
- Northeastern _____
- Information _____
- First _____
- Oliver _____
- Take me out to the _____ _____

depends on culture/context

Language Models (LM) .. 1

- *Unigram language model*
  - Probability distribution over the words in a language
  - If document treated as a sequence of words, can predict the next word given current word
  - E.g. words girl, boy, saw, dog, the may have distribution (0.25, 0.2, 0.1, 0.1, 0.35)

Probability of any word by itself: girl : 0.25

Probability of sequence girl dog: 0.25 x 0.1

Equation - probability of each word in the sequence multipled by each other.

Language Models .. 2

N-gram language model

- Some applications use bigram (N=2) and trigram (N=3) language models where
  - probabilities depend on previous words
  - Can predict next word based on previous 2 (or 3) words if bigram (or trigram)

Our focus: Unigram models

Language Models .. 3

A *topic* in a document or query can be represented as a language model

i.e., words that *tend to occur often* when discussing the topic will have high probabilities in the corresponding language model

- E.g. Document about "sailing in Alaska" will have high probabilities for words related to that topic

- Can also represent topic of a *query* as a language model

Language Models .. 4

Can use Language Model to generate text by pulling words out of a “bucket” according to the probability distribution, “with replacement”

(i.e. the word pulled out is noted, and put back before another word is chosen)

- bunch of words with no syntactic structure
- but models the topic the author had in mind while writing the document

Language Models .. 5

When text is modeled as a finite sequence of words,
t possible words at each point in the sequence

- *corresponds to multinomial* distribution over words
- commonly used, but not only possible distribution
- does not model *burstiness* well
  once a word is pulled from the bucket, will be pulled out often
  (e.g. word 'model' in this lecture)

LMs for Retrieval

Three possibilities for retrieval models based on LMs:

1. Query from Doc: probability of **generating** the **query** text **from** a **document** language model

2. Doc from Query: probability of **generating** the **document** text **from** a **query** language model (e.g. via relevance feedback)

3. Query vs. Doc Models: **comparing** the language models representing the **query and document** topics

QUERY LIKELIHOOD RANKING

Query-Likelihood Model

- Rank documents by the probability that the query text could be generated by the document model
  - i.e. probability that we can pull out query words out of "bucket" of words from document LM
- Given query, start with P(D|Q) to rank D

Trying to figure out what is the probability, given a query of finding this document

- Using Bayes' Rule $p(D|Q) \overset{rank}{=} P(Q|D)P(D)$

Rank equivalence - the order you sort the values will be the same between two different functions. They are essentially the same types of numbers, but they are different by some constant.

- Assuming prior P(D) is uniform, and same for all documents, and assuming a unigram model, we can rank by

$$P(Q|D) = \prod_{i=1}^{n} P(q_i|D)$$

Product of

Estimating Probabilities

- Estimate for unigram probabilities:

$$P(q_i|D) = \frac{f_{q_i,D}}{|D|}$$

Frequency of the word being queried

Number of words in the document

[$f_{qi,D}$ = #times q_i occurs in ~~doc D, |D|~~ = length of doc D]

- *Maximum likelihood estimate*
  - makes the observed value of $f_{q;D}$ most likely
- But: If any query words are missing from document, $f_{qi,D}$ will be zero, and score P(Q |D) will be zero
  - Not good for long queries, one missing word leads to zero score
  - Also: Missing 1 out of 4 query words same as missing 3 out of 4
  - Not reasonable; words associated with topic must have *some* probability of occurrence

Smoothing

Document texts are a *sample* from the language model

We do not always have large amounts of text

Missing words ***should not have zero probability*** of occurring

Smoothing: technique for estimating probabilities for missing (or unseen) words

lower (or *discount*) the probability estimates for words that are seen in the document text

assign that "left-over" probability to the estimates for words not seen in the text

Estimating Probabilities

Estimate for unseen words is $\alpha_D P(q_i|C)$

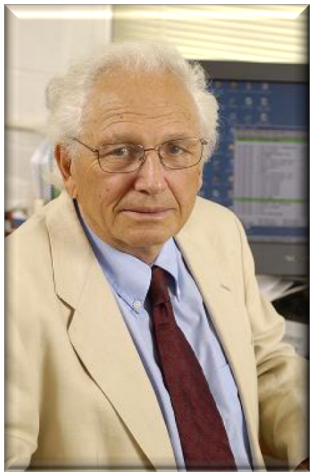
- $P(q_i|C)$ is the probability for query word i in the *collection* language model for collection C (background probability)
- α_D : parameter controlling probability assigned to unknown words

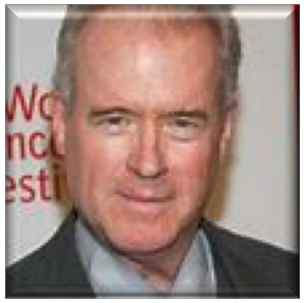
Estimate for words that occur in document D is

- $(1 - \alpha_D)\, P(q_i|D) + \alpha_D P(q_i|C)$

Different forms of estimation come from different α_D

Jelinek-Mercer Smoothing





- *If we set* α_D to a constant, λ
- Gives estimate of

$$p(q_i|D) = (1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|}$$

collection model probability estimate for q_i

- Ranking score

$$P(Q|D) = \prod_{i=1}^{n}((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})$$

- Use logs for convenience
  - accuracy problems multiplying small numbers

$$\log P(Q|D) = \sum_{i=1}^{n}\log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})$$

Fred Jelinek: From http://www.clsp.jhu.edu/~jelinek/
Bob Mercer: https://thenetworthportal.com/

Value of λ

$$\log P(Q|D) = \sum_{i=1}^{n} \log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})$$

- If λ is small,
  - Less smoothing, behaves more like Boolean AND
  - if even one query word absent, affects score significantly

- If λ approaches 1,
  - acts more like Boolean OR
  - like a “coordination level match” (# of query terms in doc)

- In TREC, λ is

  around 0.1 for short queries, 0.7 for longer queries

Where is *tf.idf* weight?

$$\log P(Q|D) = \sum_{i=1}^{n} \log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})$$

Words not in docs (unknown)

$$= \sum_{i:f_{q_i,D}>0} \log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|}) + \sum_{i:f_{q_i,D}=0} \log(\lambda\frac{c_{q_i}}{|C|})$$

Words occurring in docs , f > 0

$$= \sum_{i:f_{q_i,D}>0} \log \frac{((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})}{\lambda\frac{c_{q_i}}{|C|}} + \sum_{i=1}^{n} \log(\lambda\frac{c_{q_i}}{|C|})$$

Same for all docs, ignore

$$\overset{rank}{=} \sum_{i:f_{q_i,D}>0} \log\left(\frac{((1-\lambda)\frac{f_{q_i,D}}{|D|}}{\lambda\frac{c_{q_i}}{|C|}} + 1\right)$$

proportional to the term frequency,
inversely proportional to the collection frequency

Another smoothing method: Dirichlet Smoothing

- Let α_D depend on document length: $\alpha_D = \frac{\mu}{|D|+\mu}$
- Substituting in $p(q_i|D) = (1 - \alpha_D) P(q_i|D) + \alpha_D P(q_i|C)$, we get a probability estimation of

$$p(q_i|D) = \frac{f_{q_i,D} + \mu \frac{c_{q_i}}{|C|}}{|D|+\mu}$$

- And document score (taking products and logs):

$$\log P(Q|D) = \sum_{i=1}^{n} \log \frac{f_{q_i,D} + \mu \frac{c_{q_i}}{|C|}}{|D|+\mu}$$

- More effective, especially for short queries
- Good values for μ (from TREC): 1000 -- 2000

https://en.wikipedia.org/wiki/Peter_Gustav_Lejeune_Dirichlet

Query Likelihood Example

For the term “president” $f_{qi,D} = 15$, $c_{qi} = 160{,}000$

For the term “lincoln” $f_{qi,D} = 25$, $c_{qi} = 2{,}400$

Assume number of word occurrences in the document |d| to be 1,800

Assume number of word occurrences in the collection is 10^9 — 500,000 documents times an average of 2,000 words

Finally, choose μ = 2,000

Query Likelihood Example

$$\sum_{i=1}^{n} \log \frac{f_{q_i,D} + \mu \frac{c_{q_i}}{|C|}}{|D| + \mu}$$

$$\begin{aligned} QL(Q,D) &= \log \frac{15 + 2000 \times (1.6 \times 10^5 / 10^9)}{1800 + 2000} \\ &\quad + \log \frac{25 + 2000 \times (2400/10^9)}{1800 + 2000} \\ &= \log(15.32/3800) + \log(25.005/3800) \\ &= -5.51 + -5.02 = -10.53 \end{aligned}$$

Negative number? Logs of fractions.

Query Likelihood Example

Frequency of "president"	Frequency of "lincoln"	QL score
15	25	-10.53
15	1	-13.75
15	0	-19.05
1	25	-12.99
0	25	-14.40

Compare with BM25 scores →

Frequency of "president"	Frequency of "lincoln"	BM25 score
15	25	20.66
15	1	12.74
15	0	5.00
1	25	18.2
0	25	15.66

Ranking almost same except for 15-1 vs 0-25

Query Likelihood: Summary

Simple probabilistic model incorporating tf

Uses probability estimation. Better understood, has formal basis

Query Likelihood outperforms BM25 if better smoothing is used.

RELEVANCE MODELS / DOCUMENT LIKELIHOOD MODELS

Relevance Models .. 1

Relevance model – language model representing information need

query and relevant documents are samples from this model

Query seen as a small sample of text generated from relevance model

One way: Query
→ Relevant documents from query
→ Estimate relevance model probabilities
→ Predict relevance of new documents

Relevance Models .. 2

P(D|R) - probability of generating the text in a document given a relevance model

document likelihood model

less effective than query likelihood due to difficulties comparing documents of different lengths

Alternative: compare relevance model with document model

Pseudo-Relevance Feedback

- Estimate relevance model from query and top-ranked documents
- Rank documents by similarity of document model to relevance model
- *Kullback-Leibler divergence* (KL-divergence): well-known measure of the difference between two probability distributions


https://en.wikipedia.org/wiki/Solomon_Kullback#/media/File:Solomon-kullback.jpg
https://s3.amazonaws.com/photos.geni.com/p5/2864/7418/534448366fc9d7b2/leibler_large.jpg

KL-Divergence .. 1

Given a ("the *true")* probability distribution *P* and another distribution *Q* that approximates *P*, KL-divergence is defined as:

$$KL(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

- Number is always positive
- Larger for distributions further apart
- Not symmetric

KL-Divergence .. 1

- So: Use negative KL-divergence for ranking.

  Assume relevance model for the query *R* is the true distribution

$$\sum_{w \in V} P(w|R) \log P(w|D) - \sum_{w \in V} P(w|R) \log P(w|R)$$

No dependence on document D, can be ignored for ranking

P(x) Q(x)

KL-Divergence .. 3

- Given a simple maximum likelihood estimate for *P(w|R),* based on the frequency in the query text, ranking score is

$$\sum_{w \in V} \frac{f_{w,Q}}{|Q|} \log P(w|D)$$

  - rank-equivalent to query likelihood score
- Query likelihood model is a special case of retrieval based on relevance model

Estimating the Relevance Model .. 1

- If query words are a sample from relevance model:
  Probability of pulling word *w* out of the relevance model "bucket" depends on *n* query words pulled out

$$P(w|R) \approx P(w|q_1 \ldots q_n)$$

- This conditional probability can be expressed as a joint probability of seeing w with query words:

$$P(w|R) \approx \frac{P(w,q_1 \ldots q_n)}{P(q_1 \ldots q_n)}$$

- Denominator is a normalizing constant

Estimating the Relevance Model .. 2

- Joint probability is

$$P(w, q_1 \dots q_n) = \sum_{D \in \mathcal{C}} p(D) P(w, q_1 \dots q_n | D)$$

- Assume

$$P(w, q_1 \dots q_n | D) = P(w|D) \prod_{i=1}^{n} P(q_i|D)$$

- which leads to

$$P(w, q_1 \dots q_n) = \sum_{D \in \mathcal{C}} P(D) P(w|D) \prod_{i=1}^{n} P(q_i|D)$$

Estimating the Relevance Model .. 3

$$P(w, q_1 \ldots q_n) = \sum_{D \in \mathcal{C}} P(D) P(w|D) \prod_{i=1}^{n} P(q_i|D)$$

- *P(D)* usually assumed to be uniform
- The product term is query likelihood score for document D
- *So: P(w, q1 . . . qn)* is simply a weighted average of the language model probabilities for *w* in a set of documents, where the weights are the query likelihood scores for those documents

Pseudo-Relevance Feedback Algorithm

1. Rank documents using the query likelihood score for query Q.
2. Select some number of the top-ranked documents to be the set $\mathcal{C}$.
3. Calculate the relevance model probabilities $P(w|R)$. $P(q_1 \dots q_n)$ is used as a normalizing constant and is calculated as

$$P(q_1 \dots q_n) = \sum_{w \in V} P(w, q_1 \dots q_n)$$

4. Rank documents again using the KL-divergence score

$$\sum_w P(w|R) \log P(w|D)$$

Note:

- Steps 1, 4: Estimate P(w|D) using Dirichlet smoothing $\log P(Q|D) = \sum_{i=1}^{n} \log \frac{f_{q_i,D} + \mu \frac{c_{q_i}}{|C|}}{|D| + \mu}$
- Step 2: Use 10-50 top docs instead of whole collection; faster, ~ same
- Step 4: Use top 10-25 words, emphasize original query words with Jelinek-Mercer smoothing

How good are relevance models?

Ranking using relevance models: one of the best pseudo-relevance feedback methods

Significantly better than query likelihood (when averaged over a number of queries)

But: inconsistent, like all pseudo-relevance feedback systems

Example from using Top 10 Docs

president lincoln	*abraham lincoln*	*fishing*	*tropical fish*
lincoln	lincoln	fish	fish
president	america	farm	tropic
room	president	salmon	japan
bedroom	faith	new	aquarium
house	guest	wild	water
white	abraham	water	species
america	new	caught	aquatic
guest	room	catch	fair
serve	christian	tag	china
bed	history	time	coral
washington	public	eat	source
old	bedroom	raise	tank
office	war	city	reef
war	politics	people	animal
long	old	fishermen	tarpon
abraham	national	boat	fishery

Example from using Top 50 Docs

president lincoln	*abraham lincoln*	*fishing*	*tropical fish*
lincoln	lincoln	fish	fish
president	president	water	tropic
america	america	catch	water
new	abraham	reef	storm
national	war	fishermen	species
great	man	river	boat
white	civil	new	sea
war	new	year	river
washington	history	time	country
clinton	two	bass	tuna
house	room	boat	world
history	booth	world	million
time	time	farm	state
center	politics	angle	time
kennedy	public	fly	japan
room	guest	trout	mile

Relevance Models Summary

Ranking by comparing model of query to model of doc using KL-divergence is a generalization of query likelihood scoring

Allows for better queries, showing importance of words in topic

Relevance model estimation is an effective pseudo-relevance feedback system; but apply with care!

Moving on to more than topical relevance…

RETRIEVAL MODELS & WEB SEARCH

Web Search: what's different

Most important, but not only, search application

Web search and other searches: major differences:

- Size of collection
- Connections between documents
- Range of document types
- Importance of spam
- Volume of queries
- Range of query types …

Search Taxonomy [Broder, 2002]

- *Informational*
  - Finding information about some topic which may be on one or more web pages
  - ~ Topical search
- *Navigational*
  - finding a specific web page that the user has either seen before or is assumed to exist
- *Transactional*
  - finding a site where a task such as shopping or downloading music can be performed
- Question-Answering?

Andrei Broder

Web Search: relevance features ..1

For effective navigational and transactional search, need to combine features that reflect *user relevance*

Commercial web search engines combine evidence from *hundreds* of features to generate a ranking score for a web page

page content, page metadata, anchor text, links (e.g., PageRank), and user behavior (click logs)

page metadata – e.g., “age”, how often it is updated, the URL of the page, the domain name of its site, and the amount of text content

Web Search: relevance features ..2

In TREC evaluations, most effective features for navigational search are:

- text in the title, body, and heading (h1, h2, h3, and h4) parts of the document, the anchor text of all links pointing to the document, the PageRank number, and the inlink count

Given size of Web, many pages will contain all query terms

- Ranking algorithm focuses on discriminating between these pages
- Term proximity is important

Term Proximity

- Many models have been developed
- N-grams used in commercial web search
- *Dependence model* based on inference net has been effective in TREC
- Sample Galago query generated from user query: [embryonic stem cells]

```
#weight(
        0.8 #combine(embryonic stem cells)
        0.1 #combine( #od:1(stem cells) #od:1(embryonic stem)
                      #od:1(embryonic stem cells))
        0.1 #combine( #uw:8(stem cells) #uw:8(embryonic cells)
                      #uw:8(embryonic stem) #uw:12(embryonic stem cells)))
```

Ordered window

Unordered window

Complex Galago Web Query generated from [pet therapy]

```
#weight(
        0.1 #weight( 0.6 #prior(pagerank) 0.4 #prior(inlinks))
        1.0 #weight(
            0.9 #combine(
                #weight( 1.0 pet.(anchor) 1.0 pet.(title)
                         3.0 pet.(body) 1.0 pet.(heading))
                #weight( 1.0 therapy.(anchor) 1.0 therapy.(title)
                         3.0 therapy.(body) 1.0 therapy.(heading)))
            0.1 #weight(
                1.0 #od:1(pet therapy).(anchor) 1.0 #od:1(pet therapy).(title)
                3.0 #od:1(pet therapy).(body) 1.0 #od:1(pet therapy).(heading))
            0.1 #weight(
                1.0 #uw:8(pet therapy).(anchor) 1.0 #uw:8(pet therapy).(title)
                3.0 #uw:8(pet therapy).(body) 1.0 #uw:8(pet therapy).(heading)))
        )
```

Search Engine Optimization

SEO

Understanding the relative importance of features used in search , how they can be *manipulated* to get better search rankings [But from a black-box perspective]

Typical recommendations

improve text used in the title tag

improve the text in heading tags

ensure domain name and URL contain keywords

improve the anchor text and link structure

Some techniques frowned upon by search engine companies

link farms, keyword stuffing …

Sample tool: WebPosition Gold

MACHINE LEARNING AND IR

Machine Learning and IR

Considerable interaction between these fields

Rocchio algorithm (60s) for relevance feedback: a simple learning approach

80s, 90s: learning ranking algorithms based on user feedback

2000s: text categorization

Limited by amount of training data

Web query logs have generated new wave of research

e.g., “Learning to Rank (LeToR)”

Generative vs. Discriminative .. 1

Probabilistic retrieval models presented so far are *generative models*

A generative model

assumes that documents generated from some underlying model (usually a multinomial distribution)

uses training data to estimate parameters of model

performs well with low numbers of training examples

Identifying relevant documents

probability of belonging to a class estimated using Bayes' Rule and the document model

Generative vs. Discriminative .. 2

A *discriminative* model

- estimates the probability of belonging to a class directly from the observed features of the document based on the training data
- can easily incorporate many features
- does better given enough training data

Discriminative Models for IR

- Discriminative models can be trained using explicit relevance judgments or click data in query logs
  - Click data: much cheaper, but noisy
  - Relevance judgments: harder to acquire

- Ranking Support Vector Machine

  [Thorsten Joachims, 2002]
  - uses *partial rank* information for queries
  - i.e. partial order information about which documents should be ranked higher than others

From: http://www.cs.cornell.edu/people/tj/

Ranking SVM .. 1a

- Training data is a set of <query, ranking-info> tuples: $(q_1, r_1), (q_2, r_2), \ldots, (q_n, r_n)$
  - r_i is partial rank information about query q_i
    - if document d_a should be ranked higher than d_b, then $(d_a, d_b) \in r_i$

Ranking SVM .. 1b

– partial rank information comes from

- relevance levels ($(q_1, r_1), (q_2, r_2), \ldots, (q_n, r_n)$ levels of relevance)
- click data

  e.g., If d_1, d_2 and d_3 are the documents in the first, second and third rank of the search output, and user clicked only on d_3
  → (d3, d1) and (d3, d2) will be in desired ranking for this query

Ranking SVM .. 2a

- Learning a linear ranking function $\vec{w}.\vec{d}_a$
  - where w is a weight vector that is adjusted by learning
  - d_a is the vector representation of the features of document, based on page content, metadata, anchor text, links, user behavior etc.

Ranking SVM .. 2b

- Features usually simple, less formal than in language models. For example:
  - #words in common between query and document body,
  - #words in common between query and title etc.
- Weights represent importance of features
  - learned using training data

$$\vec{w}.\vec{d} = (2, 1, 2).(2, 4, 1) = 2.2 + 1.4 + 2.1 = 10$$

Ranking SVM .. 3

- Learn *w* that satisfies as many of the following conditions as possible:

$$\begin{aligned} \forall(d_i, d_j) \in r_1 \quad &: \quad \vec{w}.\vec{d_i} > \vec{w}.\vec{d_j} \\ &\ldots \\ \forall(d_i, d_j) \in r_n \quad &: \quad \vec{w}.\vec{d_i} > \vec{w}.\vec{d_j} \end{aligned}$$

- Can be formulated as an *optimization* problem (e.g. a minimization problem)

Ranking SVM .. 4

$$
\begin{aligned}
minimize: \quad & \frac{1}{2}\vec{w}.\vec{w} + C\sum \xi_{i,j,k} \\
subject\ to: & \\
\forall (d_i, d_j) \in r_1 \quad : \quad & \vec{w}.\vec{d_i} > \vec{w}.\vec{d_j} + 1 - \xi_{i,j,1} \\
& \ldots \\
\forall (d_i, d_j) \in r_n \quad : \quad & \vec{w}.\vec{d_i} > \vec{w}.\vec{d_j} + 1 - \xi_{i,j,n} \\
& \forall i \forall j \forall k : \xi_i, j, k \geq 0
\end{aligned}
$$

- ξ, a slack variable, allows for misclassification of difficult or noisy training examples
- *C* parameter used to prevent overfitting

Ranking SVM .. 5

- SVM Software available to do optimization
- Each pair of documents in training data can be represented by the vector: $(\vec{d}_i - \vec{d}_j)$
- Score for this pair is: $\vec{w}.(\vec{d}_i - \vec{d}_j)$
- SVM classifier will find a *w* that makes the smallest score as large as possible
  - make the differences in scores as large as possible for the pairs of documents that are hardest to rank

ML Ranking & Web Search Engines

Bing

RankNet (2005), Lambdarank (2006), LambdaMART (2008),…

Several hundred features, multi-level search judgments… [See Readings; Quiz question based on Burges, 2015]

Yandex (Russian)

ML ranking with MatrixNet since 2009

Google

"As of 2008, Google's Peter Norvig denied that their search engine exclusively relies on machine-learned ranking" (Wikipedia) 2015: Announcement about using ML in Search Ranking 2019: Announce use of BERT in search intent recognition

Apache Solr

2017: Bloomberg adds LTR plugin for Solr

Which model should we use?

Best retrieval model depends on application and data available

Evaluation corpus (or test collection), training data, and user data are all critical resources

Open source search engines can be used to find and tune effective ranking algorithms

Language resources (e.g., thesaurus) can make a big difference

Summary

Looked at language models and smoothing

Considered web search from the perspective of retrieval models

Looked at Machine Learning in IR

Finished with a brief look at topic models

Next week: **Evaluating Search Engines**

Readings

Croft, Metzler & Strohman (CMS),
Chapter 7, Sections 7.3, 7.5, 7.6, 7.7

Chris Burges, RankNet: A ranking retrospective, Microsoft Blog, 2015
https://www.microsoft.com/en-us/research/blog/ranknet-a-ranking-retrospective/ . Read this before you attempt Quiz 8!

Optional:
Manning, Raghavan & Schütze (MRS),
Chapter 12 & Section 15.4

Quiz 8

Additional Slides

TOPIC MODELS

Topic Models

Improved representations of documents

- can also be viewed as improved smoothing techniques to deal with vocabulary mismatch
- improve estimates for words that are related to the topic(s) of the document
  - instead of just using background probabilities

Approaches

- *Latent* Semantic Indexing (LSI)
- Probabilistic *Latent* Semantic Indexing (pLSI)
- *Latent* Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA)

[David Blei, Andrew Ng, Michael I Jordan, 2003]
Models document as being generated from a *mixture* of a small number of topics, each using a small set of words frequently.

1. For each document D, pick a multinomial distribution θ_D from a Dirichlet distribution with parameter α ,

2. For each word position in document D,

   (a) pick a topic z from the multinomial distribution θ_D ,

   (b) Choose a word w from $P(w|z, \beta)$, a multinomial probability conditioned on the topic z with parameter β.

http://www.cs.columbia.edu/~blei/
https://twitter.com/AndrewYNg
https://people.eecs.berkeley.edu/~jordan/

LDA

- Gives language model probabilities

$$P_{lda}(w|D) = P(w|\theta_D, \beta) = \sum_z P(w|z, \beta)P(z|\theta_D)$$

- Used to smooth the document representation by mixing them with the query likelihood probability as follows:

$$P(w|D) = \lambda \left(\frac{f_{w,D} + \mu \frac{c_w}{|C|}}{|D| + \mu} \right) + (1 - \lambda) P_{lda}(w|D)$$

LDA

If the LDA probabilities are used directly as the document representation, the effectiveness will be significantly reduced because the features are *too smoothed*

e.g., in typical TREC experiment, only 400 topics used for the *entire* collection

generating LDA topics is expensive

When used for smoothing, effectiveness is improved

LDA Example

Top words from 4 LDA topics from TREC news

Arts	*Budgets*	*Children*	*Education*
new	million	children	school
film	tax	women	students
show	program	people	schools
music	budget	child	education
movie	billion	years	teachers
play	federal	families	high
musical	year	work	public
best	spending	parents	teacher
actor	new	says	bennett
first	state	family	manigat
york	plan	welfare	namphy
opera	money	men	state
theater	programs	percent	president
actress	government	care	elementary
love	congress	life	haiti

Questions?